Remarks

Status of application

Claims 1-10, 12-37 and 39-47 were examined and claims 1-10, 12-14, 17-21 and 27-43 stand rejected in view of prior art. Applicant is grateful for the Examiner's indication of allowable subject matter in claims 15, 16, 22-26, and 44-47. Applicant has amended independent claims 1 and 29 for purposes of more clearly distinguishing them from the prior art of record. Based on the amendments to the claims and the remarks set forth below, reexamination and reconsideration of the claims are respectfully requested.

The Invention

Applicant's invention provides a cache management system providing an improved page latching methodology. Applicant's approach provides for placing a reader/writer latch is placed in a data structure associated with each page which is implemented in a manner that enables multiple threads to concurrently obtain the latch for read access without blocking one another. In addition, Applicant's invention also avoids blocking other threads when searching for a given page. Prior art systems typically require use of a mutex (mutual exclusion object) or other synchronization object when performing a search for a particular page in cache. Applicant's invention, however, avoids using a synchronization object (e.g., mutex) when searching for a page whenever possible. In its presently preferred embodiment, an indexed array (or hash table) is employed to facilitate access to pages in the cache based on the page name. In order to look up a particular page (by name), one would perform a hash of the name and find the entry in the array (hash table) corresponding to the hash value of the name. Then, the search would proceed by walking along the linked list (cache chain) looking for the particular page.

With Applicant's approach, this search for a page proceeds without any protection (i.e., it is thread unsafe). Accordingly, there is some risk that the linked list could change (e.g., pages could be inserted or removed and so forth by another thread). To address this risk, Applicant's invention includes mechanisms to avoid errors while looking for the page in this fashion and a fall back procedure if the desired page is not found through the (unprotected) search process. However, this fallback procedure is only necessary a very

small percentage of the time when the page is not in cache or another thread is in the process of updating the cache chain (linked list) that is being searched.

The advantages of Applicant's invention include that more than thread can search for and obtain access to a page concurrently and there generally is no point at which the threads have to perform serializing operations, one-at-a-time, in order to access and read the page. This means that there is no possibility of a "convoy" of threads waiting in line to perform a particular operation (e.g., acquiring a chain mutex) that is performed serially in order to read a given page. These features are particularly useful in improving access to pages that are frequently read (e.g., a page containing the root of an index), as multiple threads can read the page concurrently without having to serialize.

General

A. 35 USC Section 101 rejection

The Examiner has rejected claim 28 under 35 USC Section 101 as directed to non-statutory subject matter. Applicant has canceled claim 28.

B. 35 USC Section 112 rejection

The Examiner also rejected claim 28 under 35 USC Section 112, second paragraph as indefinite. Applicant has canceled claim 28.

Prior art rejections

A. Section 102 rejection: Oliver

The Examiner has rejected claim 27 under 35 U.S.C. 102(b) as being anticipated by US Patent 6,029,192 to Oliver (hereinafter "Oliver"). Applicant's claim 27 is a dependent claim which incorporates the limitations of independent claim 1 and adds additional claim limitations of a computer readable medium having processor-executable instructions for performing the method of claim 1. As Applicant's claim 1 includes limitations not found in Oliver as described in detail below in the discussion of the rejection of claim 1 under Section 103, Applicant respectfully believes that claim 27 overcomes the rejection under Section 102 by virtue of dependency from Applicant's independent claim 1.

B. First Section 103 rejection: Oliver

Claims 1-10, 12, 17, 20, 27, 29-37, 41, and 42, stand rejected under 35 U.S.C. 103(a) as being unpatentable over Oliver (US Patent 6,029,190). The Examiner's rejection of claims 1 and 29 as follows is representative of the rejection of Applicant's claims as unpatentable over Oliver:

> As per claims 1 and 29, Oliver substantially discloses a system for providing access to data in a multi-threaded computing system, the method comprising: providing a memory (Fig. 4, item 410) containing pages of data in memory of the multi-threaded computing system (col. 1, lines 13-16, wherein pages of data are inherent in the system of Oliver because a page is a building block that contains data in a memory);
> associating a latch/lock with each page in the memory to regulate access to the page, the latch allowing multiple threads to share access to the page for read operations and a single thread to obtain page for write operations (see Abstract, lines 8-10);
> in response to the request from a first thread to read a particular page, determining whether the particular page is in the cache without blocking access by other threads to pages in the cache, wherein said determining step, includes determining whether the particular page in the cache without acquiring a mutual exclusion object (mutex) controlling access to pages in the cache (See Fig. 1 and its corresponding description on col. 3, lines 4-44, wherein the step 110 of Fig. 1 is to determine if the object is available and it is does not acquire the object until the step 114);
> if the particular page is in the cache, attempting to obtain the latch for purposes of reading the particular page (col. 3, lines 39-43); and
> allowing the first thread to read the particular page unless a second thread has latched the particular page on an exclusive basis (col. 3, lines 22-29).
> Oliver did not explicitly disclose a cache. However, a cache is a form of memory that is taught by Oliver and it is well known in the art at that a cache allows the most frequently used data to store within until it is being replaced by another more frequently used data. Therefore, it would have been obvious to one having an ordinary skill in the art at the time of the Applicant's invention to implement a cache in the memory 410 of Oliver to take the advantage of the feature set forth in order to improve the data throughput and system performance.

Under Section 103(a), a patent may not be obtained if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which the subject matter pertains. To establish a prima facie case of obviousness under this section, the Examiner must establish: (1) that there is

some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings, (2) that there is a reasonable expectation of success, and (3) that the prior art reference (or references when combined) must teach or suggest all the claim limitations. (See e.g., MPEP 2142). As will be shown below, the Oliver reference fails to meet the requisite condition of teaching or suggesting all of Applicant's claim limitations.

Oliver describes a reader/writer lock implemented using two synchronization objects (e.g., mutex and semaphore) which permits a plurality of reader threads to access protected data simultaneously, while only allowing a single writer thread access to a protected data location (Oliver, abstract). However, Oliver is focused on a reader/writer latch that controls access to a particular data location, while Applicant's invention addresses the broader problem of enabling multiple threads to search for, access and read database pages in cache without blocking other threads, including other threads that are concurrently adding and/or deleting other pages from the cache. Applicant's invention enables more than one thread to concurrently search for and obtain access to a page in cache without having to perform serializing operations, one-at-a-time, except in relatively rare cases. This approach improves performance by significantly reducing the situations in which a "convoy" of threads must wait in line to perform particular operations serially in order to search for a given page in the cache.

As the Examiner acknowledges, Oliver does not explicitly disclose a cache for storing frequently used database pages. Thus, Oliver also does not discuss the data structures that are typically associated with a cache of this nature or the manner in which a search for a particular page in cache is conducted using these data structures. Instead, Oliver's methodology assumes that a particular data location at which protected information of interest is stored has already been located. As shown at Fig. 1 and Fig. 2, Oliver's reader/writer latch methodology starts by examining whether synchronization object(s) (mutex and/or semaphore) governing access to a particular data location are available to a given thread (Oliver column 2, lines 60-67; Figs. 1, 2). However, Oliver is silent as to how the location of the particular data of interest is determined in the first place.

With a cache used for maintaining database pages in system memory, determining whether or not a desired page is in cache is a very important consideration. Typically, the cache is not large enough to hold the entire database, and thus pages must be brought in and out of cache in response to requests for access to particular data (Applicant's specification, paragraph [0010]). Accordingly, management of the cache involves not only coordinating read and write access to the data, but also involves the process of bringing items into and out of the cache (Applicant's specification, paragraph [0010]). Additionally, cache entries are typically indexed in some fashion (e.g., hash table) to facilitate access. To search for a particular page by name, a thread might hash the name of the desired page and use the result as an index into an array to obtain a pointer to a cache or list ("cache chain") of cache entries (Applicant's specification, paragraphs [0011] -[0012]). Conventionally, at this point, the search would involve acquiring a chain mutex (or other synchronization object) before following the cache chain to locate the page in the cache chain. In a multi-threaded environment, synchronization objects (e.g., mutex) are conventionally associated with each cache chain so as to avoid problems such as two threads simultaneously writing to a page or attempting to add or remove the same page at the same time.

The problem with the conventional approach is that it impacts performance. As only one thread at a time may obtain the mutex on the cache chain, "convoys" can result when threads queue to wait for the chain mutex which must be acquired and released during the process of finding the cache entry for the page and obtaining access to it. For example, five threads may be attempting to obtain access to the page containing the index root page, but the serial nature of obtaining the mutex means that only one can do so at a time, thereby slowing down overall system performance (Applicant's specification, paragraph [0015]).

Applicant's claimed invention, however, provides instead for searching the cache chain on an unprotected basis in which no cache chain mutex or other synchronization object is acquired beforehand (Applicant's specification, paragraphs [0063]-[0065]). The search of the cache chain starting with the entry referenced by head pointer (if any) and then proceeds following the next pointer(s) in the entries. If the desired page is found in the cache chain, the thread attempts to acquire a reader/writer latch on a shared basis,

which enables the thread to read the page without blocking other threads from reading the page (Applicant's specification, paragraph [0061]). As Applicant's approach for searching for the page in cache avoids the use of a mutex or other synchronization object blocking other threads from concurrently accessing the cache chain, another thread may simultaneously access the chain to locate a page and/or during the process of adding or removing pages from the cache chain (Applicant's specification, paragraphs [0061], [0063]). Thus, Applicant's approach of performing an unprotected search (i.e., without use of a synchronization object) improves performance by enabling multiple threads to concurrently access a cache chain.

This approach of proceeding with an unprotected search of the cache chain may mean that in a small number of cases the desired page cannot be found in the cache chain (e.g., because of another thread adding or removing pages from the cache chain). Only in this circumstance in which the desired page is not found does Applicant's invention provide for use of a synchronization object (e.g., mutex) to perform the search for the page in a protected manner. Although Applicant's claims are believed to already distinguish over the art, Applicant's claims have been amended to bring these features of performing a search for a page in cache in search data structures without blocking other threads to the forefront. Applicant's amended claim 1, for instance, includes the following claim limitations:

> A method for providing access to data in a multi-threaded computing system, the method comprising:
> providing a cache containing pages of data and a search data structure mapping page names to pages in memory of the multi-threaded computing system;
> associating a latch with each page in the cache to regulate access to the page, the latch allowing multiple threads to share access to the page for read operations and a single thread to obtain exclusive access to the page for write operations;
> in response to a request from a first thread to read a particular page, determining whether the particular page is in the cache without blocking access by other threads adding or removing other pages in the search data structure and without waiting for other threads adding or removing pages from the search data structure, wherein said determining step includes determining whether the particular page is in the cache without acquiring any synchronization object controlling access to the search data structure;
>
> (Applicant's amended claim 1, emphasis added)

Applicant's invention enables more than one thread in a multi-threaded computing system to search for and obtain access to a page concurrently without requiring any serializing operations. This means that generally there is no "convoy" of threads waiting in line to perform particular operation(s) serially in order to read a given page. These features are particularly useful in improving access to pages that are frequently read (e.g., a page containing the root of an index), as multiple threads can read the page concurrently without having to serialize.

Oliver does not discuss the data structures that are typically associated with a cache for storing frequently used database pages or the manner in which a search for a particular page in cache is conducted. Accordingly, Oliver provides no teaching of performing a search for a database page in cache without blocking other threads in the manner described in Applicant's specification and claims. Therefore, as Oliver does not teach or suggest all of the claim limitations of Applicant's claims 1-10, 12, 17, 20, 27, 29-37, 41, and 42 (and other claims) it is respectfully submitted that the claims distinguish over the Oliver reference and overcome any rejection under Section 103.

C. Second Section 103 rejection: Oliver and Chauvel

Claims 13, 14, and 40 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Oliver (above) in view of Chauvel et al. (US Pub. 2002/0065992). These claims are believed to be allowable for at least the reasons cited above (as to the first Section 103 rejection) pertaining to the deficiencies of Oliver as to Applicant's invention. Chauvel does not cure these deficiencies of Oliver as it includes no teaching of performing a search for a database page in cache without blocking other threads in the manner described in Applicant's specification and claims. Accordingly, claims 13, 14 and 30 are believed to distinguish over the combined references and overcome any rejection under Section 103.

D. Third Section 103 rejection: Oliver and Parson

Claims 21 and 43 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Oliver (above) in view of Parson (US Pub. 2005/0166206). Claims 21 and 43 are dependent upon Applicant's claims 1 and 29 and are believed to be allowable for at least

the reasons discussed above regarding the deficiencies of Oliver as to Applicant's invention. Parson does not cure these deficiencies of Oliver as describes hardware queues storing data values used for resource management in a processor-based system and includes no teaching of performing a search for a database page in cache without blocking other threads in the manner described in Applicant's specification and claims. Accordingly, as the prior art reference(s), even when combined, fail to teach or suggest all the claim limitations, it is respectfully submitted that Applicant's claimed invention as set forth by these claims is distinguishable over the combined references, and that the rejection under Section 103 is overcome.

Any dependent claims not explicitly discussed are believed to be allowable by virtue of dependency from Applicant's independent claims, as discussed in detail above.

Conclusion

In view of the foregoing remarks, it is believed that all claims are now in condition for allowance. Hence, it is respectfully requested that the application be passed to issue at an early date.

If for any reason the Examiner feels that a telephone conference would in any way expedite prosecution of the subject application, the Examiner is invited to telephone the undersigned at 925 465 0361.

Respectfully submitted,

Date: August 6, 2007

/G. Mack Riddle

G. Mack Riddle, Reg. No. 55,572
Attorney of Record

925 465-0361
925 465-8143 FAX